Express Mail Label No. EV396083736US

[0001] METHOD AND SYSTEM FOR DATA RECOVERY IN A CONTINUOUS DATA PROTECTION SYSTEM

[0002] CROSS REFERENCE TO RELATED APPLICATION(S)

[0003] This application claims priority from U.S. Provisional Application No. 60/---,---, entitled "METHOD AND SYSTEM FOR CONTINUOUS DATA PROTECTION," filed on February 4, 2004, which is incorporated by reference as if fully set forth herein.

[0004] FIELD OF INVENTION

[0005] The present invention relates generally to continuous data protection, and more particularly, to data recovery in a continuous data protection system.

[0006] BACKGROUND

[0007] Hardware redundancy schemes have traditionally been used in enterprise environments to protect against component failures. Redundant arrays of independent disks (RAID) have been implemented successfully to assure continued access to data even in the event of one or more media failures (depending on the RAID Level). Unfortunately, hardware redundancy schemes are ineffective in dealing with logical data loss or corruption. For example, an accidental file deletion or virus infection is automatically replicated to all of the redundant hardware components and can neither be prevented nor recovered from by such technologies. To overcome this problem, backup technologies have traditionally been deployed to retain multiple versions of a production system over time. This allowed administrators to restore previous versions of data and to recover from data corruption.

[0008] Backup copies are generally policy-based, are tied to a periodic schedule, and reflect the state of a primary volume (i.e., a protected volume) at the particular point in time that is captured. Because backups are not made on a continuous basis,

there will be some data loss during the restoration, resulting from a gap between the time when the backup was performed and the restore point that is required. This gap can be significant in typical environments where backups are only performed once per day. In a mission-critical setting, such a data loss can be catastrophic. Beyond the potential data loss, restoring a primary volume from a backup system can be complicated and often takes many hours to complete. This additional downtime further exacerbates the problems associated with a logical data loss.

[0009] The traditional process of backing up data to tape media is time driven and time dependent. That is, a backup process typically is run at regular intervals and covers a certain period of time. For example, a full system backup may be run once a week on a weekend, and incremental backups may be run every weekday during an overnight backup window that starts after the close of business and ends before the next business day. These individual backups are then saved for a predetermined period of time, according to a retention policy. In order to conserve tape media and storage space, older backups are gradually faded out and replaced by newer backups. Further to the above example, after a full weekly backup is completed, the daily incremental backups for the preceding week may be discarded, and each weekly backup may be maintained for a few months, to be replaced by monthly backups. The daily backups are typically not all discarded on the same day. Instead, the Monday backup set is overwritten on Monday, the Tuesday backup set is overwritten on Tuesday, and so on. This ensures that a backup set is available that is within eight business hours of any corruption that may have occurred in the past week.

[0010] Despite frequent hardware failures and the necessity of ongoing maintenance and tuning, the backup creation process can be automated, while restoring data from a backup remains a manual and time-critical process. First, the appropriate backup tapes need to be located, including the latest full backup and any incremental backups made since the last full backup. In the event that only a partial restoration is required, locating the appropriate backup tape can take just as long. Once the backup tapes are located, they must be restored to the primary volume. Even

under the best of circumstances, this type of backup and restore process cannot guarantee high availability of data.

[0011] Another type of data protection involves making point in time (PIT) copies of data. A first type of PIT copy is a hardware-based PIT copy, which is a mirror of the primary volume onto a secondary volume. The main drawbacks to a hardware-based PIT copy are that the data ages quickly and that each copy takes up as much disk space as the primary volume. A software-based PIT, typically called a "snapshot," is a "picture" of a volume at the block level or a file system at the operating system level. Various types of software-based PITs exist, and most are tied to a particular platform, operating system, or file system. These snapshots also have drawbacks, including occupying additional space on the primary volume, rapid aging, and possible dependencies on data stored on the primary volume wherein data corruption on the primary volume leads to corruption of the snapshot. In addition, snapshot systems generally do not offer the flexibility in scheduling and expiring snapshots that backup software provides.

[0012] While both hardware-based and software-based PIT techniques reduce the dependency on the backup window, they still require the traditional tape-based backup and restore process to move data from disk to tape media and to manage the different versions of data. This dependency on legacy backup applications and processes is a significant drawback of these technologies. Furthermore, like traditional tape-based backup and restore processes, PIT copies are made at discrete moments in time, thereby limiting any restores that are performed to the points in time at which PIT copies have been made.

[0013] A need therefore exists for a system that combines the advantages of tape-based systems with the advantages of snapshot systems and eliminates the limitations described above.

[0014] SUMMARY

[0015] In a continuous data protection system having a primary volume and a secondary volume, a method for data recovery begins by selecting a snapshot of the primary volume to be recovered and a location on which the snapshot is to be loaded. A point in time (PIT) map is created for the selected snapshot, and the selected snapshot is loaded at the selected location. A data block from the PIT map is resolved to determine which block on the secondary volume is presented as being part of the snapshot. The snapshot is accessed via a host computer as if the snapshot was the primary volume at an earlier point in time, corresponding to the time of the selected snapshot.

[0016] A system for data recovery in a continuous data protection system includes a host computer, a primary data volume, and a secondary data volume. Creating means are used to create a snapshot of the primary data volume and storing means are used to store the snapshot on the secondary data volume. Selecting means are provided for selecting a snapshot on the secondary data volume. Accessing means are used to access the selected snapshot on the host computer, wherein the selected snapshot is presented to a user as if the selected snapshot were the primary data volume at an earlier point in time, corresponding to the time of the selected snapshot.

[0017] BRIEF DESCRIPTION OF THE DRAWING(S)

[0018] A more detailed understanding of the invention may be had from the following description of a preferred embodiment, given by way of example, and to be understood in conjunction with the accompanying drawings, wherein:

[0019] Figures 1A-1C are block diagrams showing a continuous data protection environment in accordance with the present invention;

[0020] Figure 2 is an example of a delta map in accordance with the present invention;

[0021] Figure 3 is a flowchart showing a data recovery procedure in accordance with the present invention; and

[0022] Figure 4 is a diagram illustrating a retention policy for the fading out of snapshots in accordance with the present invention.

[0023] DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

[0024] In the present invention, data is backed up continuously, allowing system administrators to pause, rewind, and replay live enterprise data streams. This moves the traditional backup methodologies into a continuous background process in which policies automatically manage the lifecycle of many generations of restore images.

[0025] System Construction

[0026] Figure 1A shows a preferred embodiment of a protected computer system 100 constructed in accordance with the present invention. A host computer 102 is connected directly to a primary data volume 104 (the primary data volume may also be referred to as the protected volume) and to a data protection system 106. The data protection system 106 manages a secondary data volume 108. The construction of the system 100 minimizes the lag time by writing directly to the primary data volume 104 and permits the data protection system 106 to focus exclusively on managing the secondary data volume 108. The management of the volumes is preferably performed using a volume manager.

[0027] A volume manager is a software module that runs on a server or intelligent storage switch to manage storage resources. Typical volume managers have the ability to aggregate blocks from multiple different physical disks into one or more virtual volumes. Applications are not aware that they are actually writing to segments of many different disks because they are presented with one large, contiguous volume. In addition to block aggregation, volume managers usually also offer software RAID functionality. For example, they are able to split the segments of the different volumes into two groups, where one group is a mirror of the other group. This is, in a preferred embodiment, the feature that the data protection system is taking advantage of when the present invention is implemented as shown in Figure 1A. In many environments,

the volume manager or host-based driver already mirrors the writes to two distinct different primary volumes for redundancy in case of a hardware failure. The present invention is configured as a tertiary mirror target in this scenario, such that the volume manager or host-based driver also sends copies of all writes to the data protection system.

[0028] It is noted that the primary data volume 104 and the secondary data volume 108 can be any type of data storage, including, but not limited to, a single disk, a disk array (such as a RAID), or a storage area network (SAN). The main difference between the primary data volume 104 and the secondary data volume 108 lies in the structure of the data stored at each location, as will be explained in detail below. It is noted that there may also be differences in terms of the technologies that are used. The primary volume 104 is typically an expensive, fast, and highly available storage subsystem, whereas the secondary volume 108 is typically cost-effective, high capacity, and comparatively slow (for example, ATA/SATA disks). Normally, the slower secondary volume cannot be used as a synchronous mirror to the high-performance primary volume, because the slower response time will have an adverse impact on the overall system performance.

[0029] The data protection system 106, however, is optimized to keep up with high-performance primary volumes. These optimizations are described in more detail below, but at a high level, random writes to the primary volume 104 are processed sequentially on the secondary volume 108. Sequential writes improve both the cache behavior and the actual volume performance of the secondary volume 108. In addition, it is possible to aggregate multiple sequential writes on the secondary volume 108, whereas this is not possible with the random writes to the primary volume 104. The present invention does not require writes to the data protection system 106 to be synchronous. However, even in the case of an asynchronous mirror, minimizing latencies is important.

[0030] Figure 1B shows an alternate embodiment of a protected computer system 120 constructed in accordance with the present invention. The host computer 102 is

directly connected to the data protection system 106, which manages both the primary data volume 104 and the secondary data volume 108. The system 120 is likely slower than the system 100 described above, because the data protection system 106 must manage both the primary data volume 104 and the secondary data volume 108. This results in a higher latency for writes to the primary volume 104 in the system 120 and lowers the available bandwidth for use. Additionally, the introduction of a new component into the primary data path is undesirable because of reliability concerns.

[0031] Figure 1C shows another alternate embodiment of a protected computer system 140 constructed in accordance with the present invention. The host computer 102 is connected to an intelligent switch 142. The switch 142 is connected to the primary data volume 104 and the data protection system 106, which in turn manages the secondary data volume 108. The switch 142 includes the ability to host applications and contains some of the functionality of the data protection system 106 in hardware, to assist in reducing system latency and improve bandwidth.

[0032] It is noted that the data protection system 106 operates in the same manner, regardless of the particular construction of the protected computer system 100, 120, 140. The major difference between these deployment options is the manner and place in which a copy of each write is obtained. To those skilled in the art it is evident that other embodiments, such as the cooperation between a switch platform and an external server, are also feasible.

[0033] Conceptual Overview

[0034] To facilitate further discussion, it is necessary to explain some fundamental concepts associated with a continuous data protection system constructed in accordance with the present invention. In practice, certain applications require continuous data protection with a block-by-block granularity, for example, to rewind individual transactions. However, the period in which such fine granularity is required is generally short (for example, two days), which is why the system can be configured to

fade out data over time. The present invention discloses data structures and methods to manage this process automatically.

[0035] The present invention keeps a log of every write made to a primary volume (a "write log") by duplicating each write and directing the copy to a cost-effective secondary volume in a sequential fashion. The resulting write log on the secondary volume can then be played back one write at a time to recover the state of the primary volume at any previous point in time. Replaying the write log one write at a time is very time consuming, particularly if a large amount of write activity has occurred since the creation of the write log. In typical recovery scenarios, it is necessary to examine how the primary volume looked like at multiple points in time before deciding which point to recover to. For example, consider a system that was infected by a virus. In order to recover from the virus, it is necessary to examine the primary volume as it was at different points in time to find the latest recovery point where the system was not yet infected by the virus. Additional data structures are needed to efficiently compare multiple potential recovery points.

[0036] Delta Maps

[0037] Delta maps provide a mechanism to efficiently recover the primary volume as it was at a particular point in time without the need to replay the write log in its entirety, one write at a time. In particular, delta maps are data structures that keep track of data changes between two points in time. These data structures can then be used to selectively play back portions of the write log such that the resulting point-in-time image is the same as if the log were played back one write at a time, starting at the beginning of the log.

[0038] Figure 2 shows a delta map 200 constructed in accordance with the present invention. While the format shown in Figure 2 is preferred, any format containing similar information may be used. For each write to a primary volume, a duplicate write is made, in sequential order, to a secondary volume. To create a mapping between the two volumes, it is preferable to have an originating entry and a

terminating entry for each write. The originating entry includes information regarding the origination of a write, while the terminating entry includes information regarding the termination of the write.

[0039] As shown in delta map 200, row 210 is an originating entry and row 220 is a terminating entry. Row 210 includes a field 212 for specifying the region of a primary volume where the first block was written, a field 214 for specifying the block offset in the region of the primary volume where the write begins, a field 216 for specifying where on the secondary volume the duplicate write (i.e., the copy of the primary volume write) begins, and a field 218 for specifying the physical device (the physical volume or disk identification) used to initiate the write. Row 220 includes a field 222 for specifying the region of the primary volume where the last block was written, a field 224 for specifying the block offset in the region of the primary volume where the write ends, a field 226 for specifying the where on the secondary volume the duplicate write ends, and a field 228. While fields 226 and 228 are provided in a terminating entry such as row 220, it is noted that field 226 is optional because this value can be calculated by subtracting the offsets of the originating entry and the terminating entry (field 226 = (field 224 - field 214) + field 216), and field 228 is not necessary since there is no physical device usage associated with termination of a write.

[0040] In a preferred embodiment, as explained above, each delta map contains a list of all blocks that were changed during the particular time period to which the delta map corresponds. That is, each delta map specifies a block region on the primary volume, the offset on the primary volume, and physical device information. It is noted, however, that other fields or a completely different mapping format may be used while still achieving the same functionality. For example, instead of dividing the primary volume into block regions, a bitmap could be kept, representing every block on the primary volume. Once the retention policy (which is set purely according to operator preference) no longer requires the restore granularity to include a certain time period, corresponding blocks are freed up, with the exception of any blocks that may still be

necessary to restore to later recovery points. Once a particular delta map expires, its block list is returned to the appropriate block allocator for re-use.

[0041] Delta maps are initially created from the write log using a map engine, and can be created in real-time, after a certain number of writes, or according to a time interval. It is noted that these are examples of ways to trigger the creation of a delta map, and that one skilled in the art could devise various other triggers. Additional delta maps may also be created as a result of a merge process (called "merged delta maps") and may be created to optimize the access and restore process. The delta maps are stored on the secondary volume and contain a mapping of the primary address space to the secondary address space. The mapping is kept in sorted order based on the primary address space.

[0042] One significant benefit of merging delta maps is a reduction in the number of delta map entries that are required. For example, when there are two writes that are adjacent to each other on the primary volume, the terminating entry for the first write can be eliminated from the merged delta map, since its location is the same as the originating entry for the second write. The delta maps and the structures created by merging maps reduces the amount of overhead required in maintaining the mapping between the primary and secondary volumes.

[0043] Data Recovery

[0044] Data is stored in a block format, and delta maps can be merged to reconstruct the full primary volume as it looked like at a particular point in time. Users need to be able to access this new volume seamlessly from their current servers. There are two ways to accomplish this at a block level. The first way is to mount the new volume (representing the primary volume at a previous point in time) to the server. The problem with this approach is that it can be a relatively complex configuration task, especially since the operation needs to be performed under time pressure and during a crisis situation, i.e., during a system outage. However, some

systems now support dynamic addition and removal of volumes, so this may not be a concern in some situations.

[0045] The second way to access the recovered primary volume is to treat the recovered volume as a piece of removable media (e.g., a CD), that is inserted into a shared removable media drive. In order to properly recover data from the primary volume at a previous point in time, an image of the primary volume is loaded onto a location on the network, each location having a separate identification known as a logical unit number (LUN). This image of the primary volume can be built by using a method 300 to recover data by accessing a previously stored snapshot, as shown in Figure 3.

[0046] The method 300 begins (step 302) by selecting a snapshot for the primary volume to be recovered (step 304). Since there will be multiple snapshots available for each protected volume, the actual snapshot which is required for access needs to be selected. A list of available snapshots for a particular protected volume can be displayed from a graphical user interface (GUI) of the data protection system. The snapshot to be selected can be either a scheduled snapshot or an any point in time (APIT) snapshot.

[0047] Figure 4 shows a diagram of a retention policy used in connection with fading out the APIT snapshots over time. The retention policy consists of several parts. One part is used to decide how large the APIT window is and another part decides when to take scheduled snapshots and for how long to retain them. Each scheduled snapshot consists of all the changes up to that point in time; over longer periods of time, each scheduled snapshot will contain the changes covering a correspondingly larger period of time, with the granularity of more frequent snapshots being unnecessary.

[0048] The user can select any time that occurs within the APIT coverage. If the selected point in time occurs within a period in which the write log is out of sync (usually due to an earlier shutdown or error condition), then APIT snapshots for that period will not be available (the user will not be able to select a point in time for which

the write log is out of sync). The list of times for which the write log is out of sync are determined by the data protection system and are saved with the primary volume.

[0049] Referring back to Figure 3, a restore LUN is selected to load the snapshot onto (step 306). The restore LUN is the method for accessing a snapshot from the host. In this role, the restore LUN acts as a virtual removable media disk device (e.g., a CD drive) and the snapshot to be accessed acts as virtual piece of removable media (e.g., a CD). It is possible to restrict access to the restore LUN, permitting only authorized host computers to access the restore LUN. This type of access can be set via an access policy or other suitable access control mechanism.

[0050] Next, a determination is made whether the selected snapshot (from step 304) is a scheduled snapshot or an APIT snapshot (step 308). If the selected snapshot is a scheduled snapshot, then a point in time (PIT) map is created for the snapshot using a delta map manager (step 310). Regions of all the delta maps prior to the time of the selected snapshot are merged to create the PIT map region by region. In order to enhance performance and the speed of access to a snapshot, the snapshot data can be accessed while the PIT map is being constructed. The snapshot is "loaded" onto the selected restore LUN (step 312), and the method terminates (step 314).

[0051] If the selected snapshot is an APIT snapshot (step 308), then a new delta map is created covering the time between the time of the selected snapshot and the time of the delta map immediately preceding the time of the selected snapshot (step 316). This new delta map is created because there is not necessarily a delta map corresponding to the time of the selected APIT snapshot. Due to the nature of APIT coverage, it is simply not feasible to store delta maps for every point in time in the APIT window. The procedure then continues with step 310, as described above, with the new delta map being used in connection with the creation of the PIT map.

[0052] When an application or file system accesses a certain block on the restore LUN, the system uses the map to determine which block should be returned. If this particular block has not been resolved yet, the block is resolved immediately. Resolving a block refers to the map merging process. When a certain block has been "resolved," it

means that through map merging it has been determined which block on the secondary volume should be presented to the host as part of the removable media. This creates the illusion to the user that the full volume has already been recreated. To avoid possible delays when accessing portions of the restore LUN, the user may request that the entire map be generated and loaded into memory. This will cause a longer delay before the first access, but creates a more predictable delay once the snapshot is mounted.

[0053] After the snapshot has been loaded onto the restore LUN, the user can access the snapshot as if it were the primary volume at the selected previous point in time. The snapshot is fully read/write accessible, and the user can perform a roll-forward of all the writes that occurred from the time of the snapshot. Changes made to the snapshot are not duplicated onto the primary volume, because the snapshot is, by definition, a reflection of the primary volume at a previous point in time. It is noted that while the user is accessing a snapshot, the primary volume is still being protected as under normal operating conditions. Furthermore, different snapshots can be loaded into different LUNs; the user is not restricted to accessing one snapshot at a time. Once the user is finished with the restore LUN(s), the GUI can be used to unload the snapshot or the snapshot can be ejected from the shared removable media drive by the host, similar to how a CD can be ejected.

[0054] Another important point to mention is that read/write access is important in this scenario. This is because when an application or even a journaled file system attempts to recover from the (possibly inconsistent) state the new volume presents, these applications need to be able to replay a log or perform other writes to the volume. A system that does not offer read/write access is extremely limited in functionality. In the present invention, writes are stored in a temporary buffer, such that the original PIT image can be loaded again in its original state if desired.

[0055] In regard to performance optimization, it is not necessary to perform all of the delta map merges before the volume is presented to the host. Instead, the volume can be presented to the host immediately. Then, as the file system at the host accesses

certain blocks, these can be resolved right away. The first time the system accesses a certain block may be slower because of this, but if the system accesses the same blocks again later, the access performance will have improved. While the host is not requesting new blocks, the system automatically continues to resolve the remaining maps. The map merging being performed in this instance relates to merging all the delta maps that are relevant to the selected PIT, to create a single map of all the blocks of the primary volume at that PIT.

[0056] Users should be able to browse files and folders and search for files with certain contents, even in the absence of a server. It is nonsensical to recover an entire 200 GB volume just to check if a specific file was already corrupted at a given point in time. The present invention is able to present volumes immediately, as discussed above. So the particular file can be examined and the remainder of the volume does not need to be resolved. But this still requires a server/file system.

[0057] The present invention also has the capability of decoding file system information and presenting the user with browsable list of files via FTP or a Web interface. This interface allows users to browse to a specific directory or file and then navigate to the previous/next (or any other) snapshot that was taken of the selected file. Only the necessary blocks will be resolved for this operation, and users are able to navigate through terabytes of data in a minimal amount of time to find the restore volume they are looking for or to just restore the file or directory they are trying to recover.

[0058] Automated searches can be performed in a similar fashion, such that the system could automatically find a certain file or content. For example, if a virus struck and corrupted the system, it is difficult to navigate many volumes by time. This is because the virus could have been there already for a long time. Executable files don't change over time, except when a virus strikes, so the system could be queried to find the point in time when the executable changed. Another useful query would be to see a list of different versions of the same file, including size and attributes. From the list, the user can immediately determine the time when the file was updated, for example,

during an all-night work session, because it will include the greatest number of changes.

[0059] While specific embodiments of the present invention have been shown and described, many modifications and variations could be made by one skilled in the art without departing from the scope of the invention. The above description serves to illustrate and not limit the particular invention in any way.

* * *